RELATÓRIO ESINF - SPRINT 4

DIAGRAMA DE CLASSES E COMPLEXIDADE DE ALGORITMOS

GRUPO 21 – 2DC

1191507 – Bárbara Pinto

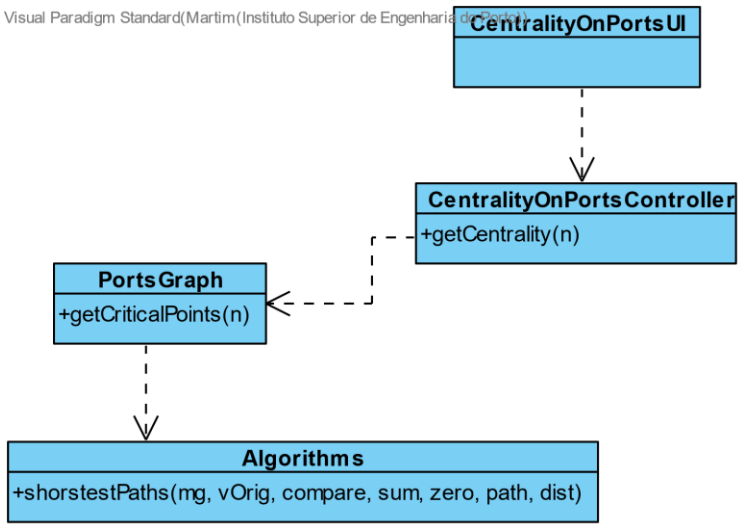
1200991 - Carlos Dias

1201029 - Cristóvão Sampaio

1201045 - Miguel Silva

1201154 – Martim Maciel

US401 – 100% - Martim Maciel 1201154



Esta *User Story* tem como objetivo descobrir quais são os portos com maior centralidade, isto é, os portos com maior número de *shorstest paths* que passam por eles. Para isto teremos de decorrer a um método que permita passar por todos os vértices do grafo de modo a obter todos os *shortest paths* possíveis, onde de seguida realizaremos o tratamento dos dados.

Assim, é criado um método como referido anteriormente onde todos os *shortest paths* para todos os vértices são obtidos. Dentro deste método podemos encontrar um *loop* inicial que irá percorrer todos os vértices do grafo, incluindo os seus *shortest paths*. Este *loop* irá identificar todos os vértices do grafo que são *Storages* e irá também guardar todos os seus *paths* numa *List* de *LinkedList*. Por fim, será inicializado um *HashMap* onde guardamos as *Storages* e o número de vezes que são percorridas. Como iremos percorrer todos os vértices do grafo iremos ter complexidade $\boldsymbol{O(V^3)}$, uma vez que todos os vértices passarão por *shortestPaths* ao qual de seguida passarão por *shortestPathsDijkstra*, no qual este método tem complexidade $\boldsymbol{O(V^2)}$.

Agora iremos percorrer a *List* usada para guardar todos os *paths* de cada vértice. Vamos dar ao uso de um *loop* para percorrer todas as *LinkedLists* guardadas em *paths*. Assim, poderemos realizar um outro *loop* para percorrer todos os elementos da *LinkedList*, deste modo poderemos observar as *Storages* que percorremos em cada *path*, podendo assim, incrementar o número de vezes que passamos em cada *Storage*, atualizando o *HashMap* anteriormente referido. Isto terá complexidade $\boldsymbol{O(V^2)}$.

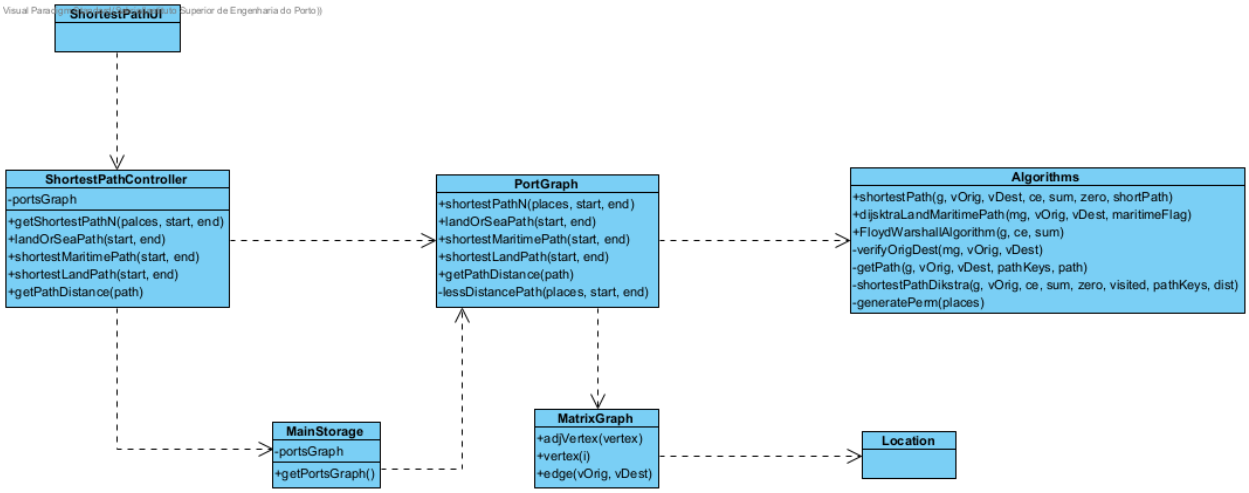
Por fim, guardados todos os valores da *HashMap* num *ArrayList* de *Pairs*, poderemos de seguida realizar um *reverseSort* à lista atual, permitindo obter assim a lista desejado dos portos com maior centralidade. De seguida realizamos um *loop* pelo número de portos solicitado pelo cliente. Isto terá complexidade de $\boldsymbol{O(V)}$.

Podemos assim concluir que esta *User Story* terá uma complexidade de $\boldsymbol{O(V^3)}$.

US402

50% - Cristóvão Sampaio 1201029

50% - Miguel Silva 1201045



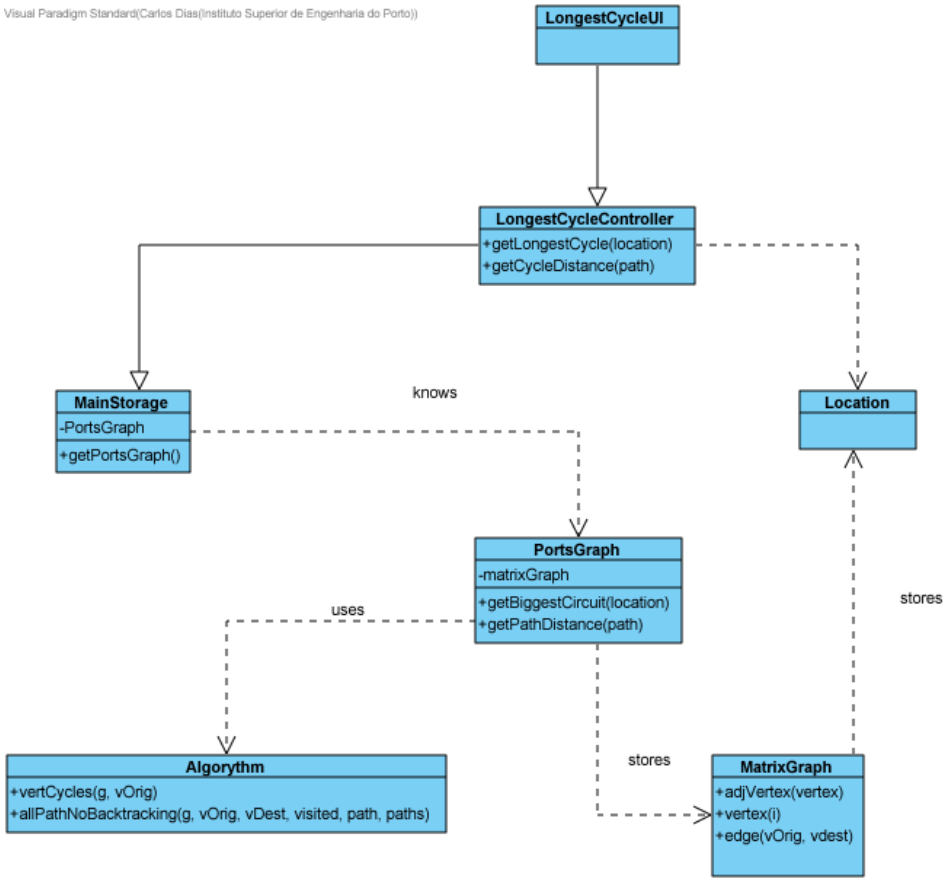
Para o caminho entre dois locais sem restrições é apenas a aplicação direta do Diijkstra que tem uma complexidade de $\boldsymbol{O(V^2)}$. O mesmo se aplica no que toca aos caminhos terrestres e marítimos, pois a única diferença é que, por exemplo nos caminhos terrestes, antes de verificar a distancia do caminho e possivelmente adicionar o vértice ao caminho, verifica se o vértice é valido ou não para o tipo de caminho.

Para descobrir o caminho mais curto entre dois locais, passando por n locais foi necessário fazer o seguinte:

Primeiro pegou se em todos os n locais obrigatórios e calculou-se todas permutações possíveis entre esses pontos intermédios, este processo tem uma complexidade de $\boldsymbol{O((V-2)!)}$. Depois utilizou se o algoritmo de Floyd-Warshall para descobrir a combinação com menos distância, ou seja, irá pegar na *edge* do vértice atual e na do vértice seguinte e por aí adiante até chegar ao fim e atualizando o caminho de menor custo se esse caminho obtido for menor que o anterior e existente também.

Por último já com o caminho feito basta apenas calcular os caminhos mínimos de cada ponto utilizando o Diijkstra que tem uma complexidade de $\boldsymbol{O(V^2)}$ mas a complexidade deste método acabara por ser $\boldsymbol{O((V-2)!)}$.

US403 – 100% - Carlos Dias 1200991



Esta User Story pede-nos para encontrar o maior ciclo, ou seja, o maior circuito que nunca repete arestas nem vértices. Isto leva-nos a um problema que se aproxima muito de um outro chamado de "O problema Hamiltoniano" que se baseia em encontrar o maior circuito que passa por todos os pontos do grafo uma única vez. Este problema é considerado de NP-Hard, ou seja, não pode ser traduzido numa equação temporal polinomial, mas sim numa equação temporal exponencial.

Este bloco não seria um problema para grafos pequenos, mas o grafo que usamos tem uns 150 vértices logo qualquer solução teria uma complexidade O(N!) o que seria impossível de computar para um grafo deste tamanho.

Assim recorremos a uma heurística para obter o que nos é pedido de uma forma muito menos precisa, mas muito mais razoável em termos de espera.

A heurística usada implica uma versão modificada do método allPaths. Neste método o backtracking é retirado e a entrada dentro nos vértices adjacentes é modificada de forma a priorizar os vértices, mas próximos. Sempre que este método se encontrar com um vértice que consegue fechar um circuito, o caminho desse circuito e guardado. No fim é extraído uma lista com todos os circuitos encontrados neste método que como só entra nos vértices ainda não visitados e ordena os vértices adjacentes com um algoritmo **O(Vlog n) tem** uma complexidade de **O(V^2Log n)**

Porem esta iteração acaba por ser bastante imprecisa. Para combater esta imprecisão, este método é corrido por todos os vértices do grafo, obtendo assim uma grande amostra de todos os circuitos do grafo, tendo uma complexidade de **O(V^3Log n).**

Finalmente, obtendo todos estes circuitos removemos todos os que não possuem o vértice de início pedido numa função de complexidade O(V^2). E com esta lista procuramos o maior número de arestas percorridas e por um caminho, e se existirem vários circuitos com o mesmo número de vértices escolhemos o que o menor tamanho percorrido, ambos este método tem complexidade de O(V). No caso de o ciclo escolhido não começar no vértice escolhido, este é rodado até o vértice escolhido ser o inicial

Desta forma a complexidade final desta User Story é **O(V^3Log n).**